LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-ID-151762

# Cooperative Mobile Sensing Networks

*R. S. Roberts, C. A. Kent, E. D.Jones, C. T. Cunningham, G. W. Armstrong*

**February 10, 2003**

# Cooperative Mobile Sensing Networks

Randy S. Roberts, Claudia A. Kent, Erik D. Jones,
Christopher T. Cunningham and Gary W. Armstrong

Lawrence Livermore National Laboratory,
Livermore, CA 94550 USA

## ABSTRACT

*A cooperative control architecture is presented that allows a fleet of Unmanned Air Vehicles (UAVs) to collect data in a parallel, coordinated and optimal manner. The architecture is designed to react to a set of unpredictable events thereby allowing data collection to continue in an optimal manner.*

**Keywords:** Unmanned Air Vehicles, Robot Cooperation, Automated Sensing, Distributed Sensing

## 1. INTRODUCTION

With the development of low cost, high endurance Unmanned Air Vehicles (UAVs), it is now practical to perform autonomous sensing and data collection over broad areas. These vehicles offer the ability to sense the environment directly through on-board sensors, or vicariously through sensors placed within the region. The advantages of using multiple UAVs to collect sensor data are manifold. Multiple vehicles allow sensing to be performed in parallel, thereby reducing the amount of time required to gather data. If a vehicle becomes disabled, the remaining vehicles can continue sensing, albeit at a reduced collection rate. Sensing with multiple UAVs has many applications of interest including providing communication services to isolated ground sensors, aerial monitoring of national borders and seaways for homeland security, and air sampling for atmospheric modeling.

This project investigated and solved two fundamental problems associated with using a fleet of UAVs to autonomously collect sensor data. The first problem is that of constructing efficient routes that allow the UAVs to collect data without duplicating effort or interfering with one another. This particular type of routing problem is fundamentally a combinatorial optimization problem. Our investigation developed a heuristic solution to this problem, and it is reported in [1]. Essentially, the approach described in [1] is to consider a set of waypoints that describe where the UAVs are to collect data, and assign a cost to each link between pairs of waypoints. An initial set of contiguous, non-overlapping routes is then constructed, one for each UAV. The global cost of this route structure is optimized by swap links between UAV routes to obtain a global minimum.

The second problem is that of devising control schemes for individual UAVs that collectively allow a fleet of UAVs to adapt to exceptional events in a globally optimal manner [2]. Such events include the loss or addition of UAVs to the network or changes in the sensing requirements such as increased prioritization of particular subregions. Our investigation resulted in several control architectures based on hierarchical or distributed control of network adaptation, and variations in the route optimization algorithm developed in [1]. Essential to these architectures is a cost associated with the sensing task performed by each UAV (a generalization of the cost metric used in [1]). This cost information is continually estimated by each UAV, and shared throughout the network. In the hierarchical architectures, one UAV is selected to lead network adaptation. This leader is not unique, and can be replaced by any other UAV should the leader leave the network. This type of architecture is useful in small networks where adaptation needs to be performed quickly. In the distributed architectures, UAVs cooperatively perform local network adaptation by locally

optimizing the costs of their subnets. These architectures are useful in large networks where local sensing costs can change rapidly.

In order to further our investigation, we have developed a software tool called STOMP (Simulation, Tactical Operations and Mission Planning). STOMP is a software architecture for simulating, controlling and communicating with UAVs employed in broad area sensing applications [3]. It implements the control architectures described in [2], and provides hardware-in-the-loop simulation capability enabling real UAVs and sensors to interacting with virtual UAVs and sensors residing in STOMP.

Finally, we conducted several experiments with Laboratory UAVs where we collected imagery from unattended ground sensors via an overhead UAV, and transmitted the data to a groundstation. These experiments allowed us to test the communications techniques developed earlier in the project, and apply the results to the design of the STOMP communications objects.

## 2. UAV ROUTE PLANNING

An issue of fundamental importance in using UAVs to conduct sensing missions is route planning. We devised a routing algorithm that allows a fleet of UAVs to optimally service waypoints while minimizing interference and duplication of effort. For the moment, we consider a static route structure. Adaptation of the route structure is considered later.

Formally stated, we are given $N$ waypoints and $K$ UAVs, and require a path planning algorithm that generates $K$ non-overlapping, non-branching, closed paths to every waypoint in the network. The route network can be modelled as a family of graphs $\{(S_k, P_k)\}$, $k = 1, \ldots, K$ where $S_k = \{s_i\}_k, i = 1, \ldots, N_k$ is the set of waypoints in the $k^{th}$ subnet, and $P_k = \{l_{ij}\}$ is the set of weighted, directed links that connect waypoint $s_i$ to waypoint $s_j$. Weight $c_{ij}$ associated with link $l_{ij}$ is the cost of the link between waypoints $s_i$ and $s_j$. The cost of path $P_k$, denoted as $C_k$, is the sum of the weights of the links in $P_k$.

This particular route planning algorithm can be formulated as a variant of an $M$-ary Traveling Salesman problem [4]. Given the $N$ waypoints and $K$ UAVs, we seek as assignment array $x_{ij}^k$, $x \in \{0, 1\}$ that will minimize cost function $Z(t)$ where

$$Z(t) = \sum_i \sum_j \sum_k c_{ij}(t) x_{ij}^k \tag{1}$$

subject to the following constraints:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} x_{ij}^k = 1, j = 1, \ldots, N \tag{2}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K} x_{ij}^k = 1, i = 1, \ldots, N \tag{3}$$

and

$$\sum_{j=1}^{N} x_{ip}^k - \sum_{j=1}^{N} x_{pj}^k = 0, k = 1, \ldots, Kp = 1, \ldots, N \tag{4}$$

The first two constraints assure that every waypoint is visited by only one UAV, and the third constraint provides for the continuity of routes. One final constraint is required for a formal definition of the problem. This constraint eliminates subtours, and is described in [4].

Equation (1) with the constraints (2)— (4) is a formidable combinatorial optimization problem. A heuristic solution to the UAV route planning algorithm was developed in this project, and is

reported in [1] with advancements to the algorithm reported in [2]. The basis for the path planning algorithm is the cost function

$$C = \sum_{k=1}^{K} (C_k)^a \tag{5}$$

where the individual terms $C_k$ in the summation are the costs that each path contributes to the total cost of the network. The exponent $a$ provides a parametric means to combine the desires for minimum total cost and roughly equivalent individual costs. Values in the range of $3 \leq a \leq 6$ have been found to work well in practice. The heuristic solution is generated in three steps: 1) an initialization step which clusters waypoints into groups, 2) an initial path plan for the clusters, and 3) a balancing operation that tries to minimize the global cost of the routing plan. An overview of the path planning algorithm is described next.

As reported in [1], first step of the initialization process is to construct $K$ subnets of waypoints $S_k$, from the $N$ waypoints in the network. The procedure is based on the $K$-means algorithm with one notable exception. A $K$-means algorithm begins by randomly selecting $K$ waypoint positions from a set, and using these positions as the initial centroids of $K$ clusters [7]. Clusters are formed by assigning each waypoint in the set to the nearest centroid. A new centroid is computed as the average over all waypoints in the cluster. This process continues until the $K$ centroids are fixed. As noted in [1], this type of $K$-means algorithm did not produce acceptable results. Instead, a modified algorithm is used that initializes by finding $K$ widely separated waypoints. This algorithm maximizes the minimum distance between any two waypoints in a set of $K$ waypoints. After these initial waypoints have been determined, the $K$-means algorithm proceeds in the customary manner.

After the initial subnets have been formed, paths are constructed to connect the waypoints in the subnet. Path $P_k$ through the $k^{th}$ subnet is constructed from a circumferential path around $S_k$, which forms the initial path for the subnet. The circumferential path around $S_k$ is the convex hull of $S_k$. As the convex hull computation progresses, an ordered list of waypoints is produced that when linked form a circumferential path around the subnet [8]. Denote the set of waypoints that form the convex hull of $S_k$ as $H$. Observe that $H$ partitions the subnet into two groups of waypoints: those on $P$, and those interior to $P$. Denote the set of waypoints in the interior of the subnet as $Q = S_k - H$. Waypoints $s_q \in Q$ are added to path $P$ in positions that minimize their contribution to the global cost (5). The differential cost of adding waypoint $s_q$ to the path between waypoints $s_i$ and $s_j$ is found by breaking link $l_{ij}$ into links $l_{iq}$ and $l_{qj}$, and is given by

$$\Delta c_{iqj} = c_{iq} + c_{qj} - c_{ij} \tag{6}$$

Waypoints in $Q$ are inserted into path $P$ at the position that minimizes (6). The process of adding waypoints in $Q$ to the path continues in this manner until all waypoints in the subnet have been assigned a position in the path.

After the initial paths to all waypoints in the network have been computed, the route structure of the overall network is balanced (optimized) by shifting waypoints between subnets. Consider paths $p$ and $p'$ in two neighboring subnets. The differential cost of delinking waypoint $s_j$ from waypoints $s_i$ and $s_k$ in path $p$ and inserting it into the link between waypoints $s_m$ and $s_n$ in path $p'$ is given by (cf. (5))

$$\Delta C = \Delta C_p + \Delta C_{p'} \tag{7}$$

where

$$\Delta C_p = (C_p - \Delta c_{ijk})^a - (C_p)^a \tag{8}$$

and

$$\Delta C_{p'} = (C_{p'} + \Delta c_{mjn})^a - (C_{p'})^a \tag{9}$$

If a particular combination of $j$, $\{i,k\}$, $\{m,n\}$, and $\{p,p'\}$ yields a $\Delta C < 0$, then the global path cost will decrease if the move is performed. By testing all waypoints in all links of all paths in the

network, and moving only those waypoints that decrease the global path cost, an optimal path (and subnet) configuration is obtained.

For a fixed network cost structure, i.e., where $c_{ij}(t)$ is constant, the UAV paths can be computed using the algorithm described above. In that algorithm, the cost was the *length* of the path though the subnet. In [2], the cost was the amount of *time* required for a UAV to traverse the subnet. Thus, element $c_{ij}$ of the cost matrix is given by

$$c_{ij}(t) = \tau_{ij} + \Delta t_i \tag{10}$$

where $\tau_{ij}$ is the amount of time required for a UAV to traverse from the $i^{th}$ sensor to the $j^{th}$ sensor and $\Delta t_i$ is the amount of time that the UAV dwells at the $i^{th}$ sensor. This cost metric is more relevant to the problem of collecting data with UAVs. For instance, headwinds and other disturbances can greatly increase the time it takes for a UAV to collect data in its subnet. As the amount of time that it takes to collect data increases, the cost of the data collection increases and the UAVs can react by spreading the increased cost over the network.

The quantities $\tau_{ij}$ and $\Delta t_i$ are random quantities, and are thus required to be estimated by the UAVs. Prior to a UAV flying between waypoints $s_i$ and $s_j$ and collecting data at waypoint $s_j$, we can only guess at values for $\tau_{ij}$ and $\Delta t_i$. After a UAV has flown this route and collected data, we can measure the amount of time that has transpired. To accomodate both situations, we use two techniques to estimate the cost of the link. Denote the distance between $s_i$ and $s_j$ as $d_{ij}$ and the maximum speed and acceleration of the UAV as $V_{max}$ and $A_{max}$. If the waypoints are spaced such that the UAV has time to accelerate, cruise and decelerate without overshooting the sensor, then the travel time can be approximated as

$$\tau_{ij} \approx 2\frac{V_m}{A_m} + \left(\frac{d_{ij}}{V_m} - \frac{V_m}{A_m}\right) \tag{11}$$

After a UAV has traveled the link, the travel times are measured and filtered to yield estimates $\hat{\tau}_{ij}$. The dwell time measurements are processed in a similar manner to yield estimates $\Delta\hat{t}_i$. The cost estimates are given by either the preliminary distance-based estimates, or the filtered measurement-based estimates:

$$\hat{c}_{ij} = \begin{cases} \hat{\tau}_{ij} + \Delta\hat{t}_i, & \hat{\tau}_{ij} \neq 0 \\ \tau_{ij} + \Delta\hat{t}_i, & \text{otherwise} \end{cases} \tag{12}$$

Finally, the cost of the subnet is given by

$$\hat{C}_k = \sum_{i=1}^{N_k-1} \hat{c}_{i,i+1} + \hat{c}_{N,1} \tag{13}$$

An example of the routes produced by this algorithm is illustrated in Figure 1. That figure shows the route plan produced for fourteen UAV and eighty waypoints. Other examples of the route planning algorithm are shown in Figure 3.

## 3. UAV AUTOMATION

The path planning algorithm previously described is suitable for relatively static scenarios where the number of UAVs and waypoints does not change, and the costs associated with collecting data remain relatively constant. However, in many scenarios of interest the number of UAVs and waypoints change as well as the cost of collecting data. In these situations we desire an automation architecture that can adapt the network routing structure as conditions warrant. Recall that the data collection task is defined by a set of waypoints, i.e., spatial coordinates, where a UAV collects data. Data collection can be performed by operating onboard sensors, or by uplinking data from sensors placed within a region.

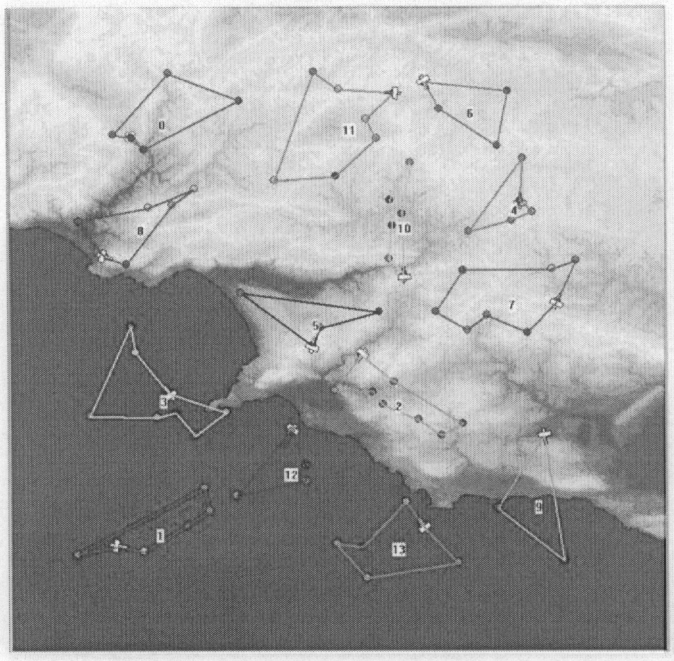

**Figure 1**. Illustration of route planning for 14 UAVs and 80 waypoints

With adaptability in mind, an automation architecture for collecting data with a fleet of UAVS was developed by this project. The architecture is designed to adapt the network route structure to the following exceptional events:

1. UAVs are added or removed from the network
2. Waypoints are added or removed from the network
3. Waypoints change their positions or priorities within the network

These events are fundamental changes to the data collection architecture. More complicated scenarios can be decomposed into these fundamental events. Thus, the adaptation algorithms were designed to react to these core events. The full details of the UAV automation algorithms are detailed in LLNL Record of Invention IL-11106, (see [9]). Here, we give an overview of the UAV automation architecture. Variations of the architecture described here, along with performance characteristics, are described in [2].

Network adaptation can be driven either globally, locally or by several hybrid schemes [2]. The UAVs share information they gather on the state of waypoints in the network as well as their own status. Since all UAVs have similar views of the state of the network, any one UAV can direct global adaptation. Such adaptation is required if, for example, a UAV leaves the network. In this case, a single UAV recomputes the new route assignments for all remaining UAVs in the network, and instructs them to begin executing the new routes. On the other hand, for minor network adjustments, such as a ground sensor moving between two subnets, the involved UAVs perform a peer-to-peer transaction where the moving sensor is transferred from one subnet to the other.

The UAV automation architecture that implements these adaptations is implemented as a set

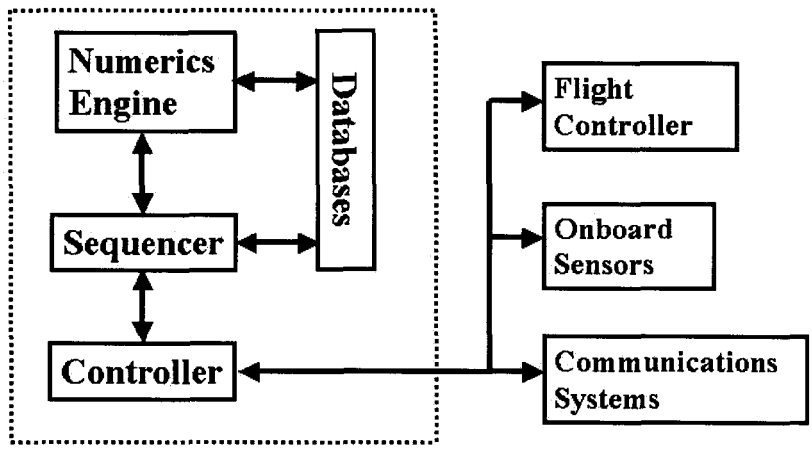


**Figure 2**. Block diagram of the UAV Automation Architecture

of algorithmic modules and databases that reside and execute on a control computer onboard each UAV in the network. It interfaces with the UAV flight controls and instrumentation, the UAV communications system, and any onboard sensors as illustrated in Figure 2.

The architecture is based on a three-component approach similar to those described in [10]. The first component, called the sequencer, is a looping structure that the UAV control computer continually executes. The sequencer, through the controller, takes inputs from various UAV subsystems including flight controls and communications. Depending on the state of these inputs, the sequencer branches to different routines designed to react to changing states. The second portion of the architecture is the Numerics Engine. The sequencer invokes the Numerics Engine to perform path planning and related calculations. The final portion of the architecture is the Controller, which interfaces to the UAV flight controls, communications and any onboard sensors. In the sequel, we focus on the sequencer modules and supporting databases.

The sequencer consists several algorithmic modules and two databases. The algorithmic components of the sequencer include: 1) Waypoint Service Routine, 2) Message Handler Routine, 3) Network Adaptation Routine, 4) Cost Estimator, 5) Sensor Service Routine, 6) Network Leadership Routine, 7) Local Route Optimizer, and 8) Waypoint Sequence Routine. The databases contain information relating to the operation of the network. One database contains data related to all waypoints in the network. This data includes, *inter alia*, such information as a waypoint identifier, waypoint state information, and time of last UAV visit. The second database contains information related to the UAVs in the network. This database includes, *inter alia*, unique identifiers for each UAV and the position of each UAV in leadership succession. The functions of the sequencer modules are briefly described below.

The Waypoint Service Routine performs data collection tasks specific to each waypoint. Data collection tasks at each waypoint can vary, depending on the nature of the sensor. Such tasks might

include uplinking data from a ground sensor, or dwelling over a region to observe an event with onboard video cameras. Similarly, the Sensor Service Routine operates any on-board sensors.

The Message Handler Routine (MHR) processes status messages from other UAVs in the network. In particular, two types of messages are processed by MHR, a waypoint status message (WSM) and UAV status message (USM). As the UAVs collect data at waypoints, they broadcast a WSM to all UAVs in the network. This message contains a variety of information related to the state of the waypoint. As a result, each UAV has a complete view of all waypoints in the network. Using this information, the Cost Estimator estimates the cost of each link in the network. This cost is updated each time a UAV receives a new WSM. In this manner each UAV can track the cost of every subnet in the network. The WSM is also used to broadcast the addition or deletion of waypoints from the network.

Using UAV status messages, each UAV has knowledge of all UAVs in the network. If any UAV intentionally leaves or enters the network, the other UAVs are informed via a USM. If any UAV unintentionally leaves the network, a second mechanism is used. The UAV database contains a field that indicates the health of each UAV in the network. The health of a UAV is determined by its appearance in the network routing tables. If a UAV appears in the table, it is assumed that the UAV can communicate and is capable of carrying out its tasks. The Network Leadership Routine, in conjunction with UAV status messages and UAV heartbeat, determines whether the leader is functioning. If not, leadership is transferred to another UAV. The Network Adaptation Routine on the leader UAV monitors the various status messages, and executes a global network reorganization if warranted.

The Local Route Optimizer (LRO) and Waypoint Sequence Routine (WSR) perform local optimizations. The LRO monitors costs in neighboring subnets, and initiates a local reorganization if subnet costs are not in balance. The WSR determines the starting waypoint of a UAV's traverse of its subnet. The starting point of the traverse is found as the waypoint that minimizing the maximum amount of time that any waypoint in the subnet must wait for the UAV.

The automation algorithms described here were tested using several thousand Monte Carlo simulations on random networks. The results of these simulations suggest that global reorganizations can be performed more quickly and optimally for small networks than purely local techniques. However, the global methods do not scale for larger networks where localized optimizations might have an advantage. See [2] for more details on the simulations are results.

## 4. SIMULATION, TACTICAL OPERATIONS AND MISSION PLANNING

In conjunction with the development of the route planning and automation algorithms, it became apparent that additional simulation tools were required to fully understand UAV-based sensing and data collection. Moreover, there was a desire to incorporate real hardware into the simulation, so that more realistic simulations could be performed and evaluated. The result of this desire is STOMP, for Simulation, Tactical Operations and Mission Planning [3]. STOMP is an application and framework designed to study and operate sensor networks where UAVs are fundamental to the collection of data.

STOMP is designed to simulate UAVs, sensors and their interactions in a distributed sensor network under a variety of conditions. It is designed to easily implement control and cooperation architectures, and displays the reaction of the architecture to events that the designer can script into the simulation through a graphical interface. Through an internal communication controller, STOMP can feedback information from real UAVs and sensors using wireless Ethernet for data acquisition from sensors and a wireless serial interface for command, control and state information thereby providing hardware-in-the-loop simulations. Thus, command, control and state information can be exchanged with real UAVs and sensors, creating a feedback mechanism to both test new

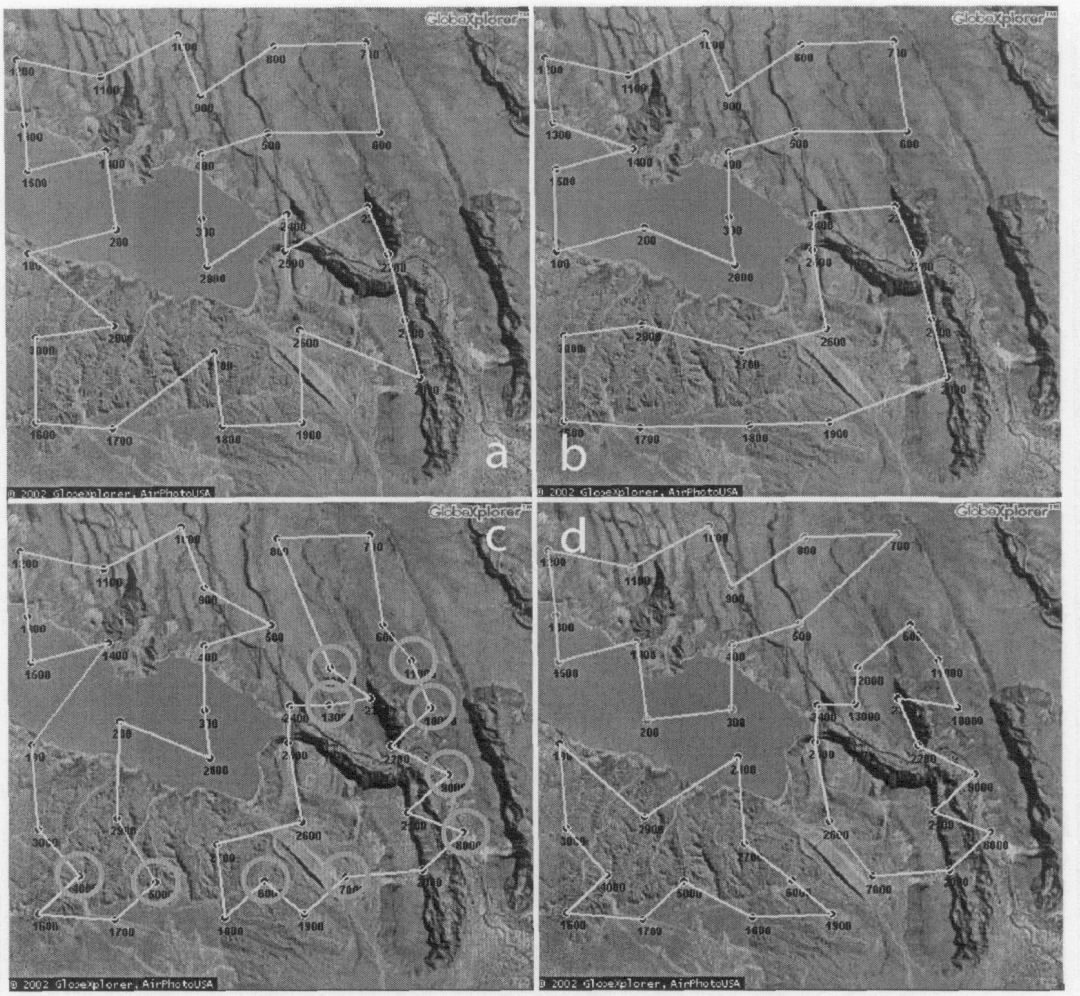


**Figure 3.** Four panels illustrating network adaptation. (a) Route of one UAV servicing sensors around a dam. (b) A second UAV is added to the network. (c) Several more sensors are added to the network, and the UAVs adjust their paths to accomodate the new sensors. (d) A third UAV is added to the network.

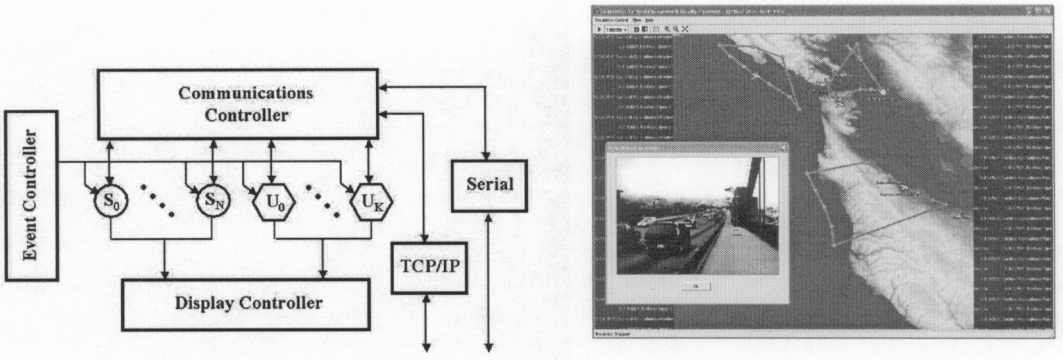


**Figure 4**. Functional Block Diagram of STOMP and STOMP console display

algorithms within the virtual simulation as well as hardware and software implementation in real UAVs and sensors.

A functional block diagram of STOMP is illustrated in Figure 4. As shown in the figure, STOMP consists of the following main blocks: 1) Sensor and UAV objects (where sensor objects are depicted as circles, and UAV objects are depicted as hexagons); 2) a Communication Controller; 3) an Event Controller; and 4) a Display Controller. These objects are briefly described below. See [3] for a comprehensive description.

The framework for STOMP is designed around an object oriented architecture thereby allowing designers to adapt behaviors and algorithms of existing objects or assemble new objects rapidly and easily. As in [11], STOMP uses a visual editor to allow designers to assemble and configure each simulation. Designers may specify the state of every object in the system individually, in groups or globally and, using the graphical event configuration system, define events to trigger state changes at specified times in order to test complex behaviors and response to exceptions.

STOMP is divided into two different display modes, the designer view and the simulator view. When a new simulation is being created, the designer view is used to place objects within the coordinate space of the DTED space represented as a color coded topographical relief map. The user may either input the coordinates directly or use the mouse and graphical interface to place objects. While in the designer view, object properties and initial states may also be set.

When the simulation is started, the simulator view appears in front of the designer view. In this mode, display controller provides the designer with visual feedback of the state of the network as it progresses. It also displays data collected by operational UAVs from deployed sensors. The positions of the UAVs and sensors, along with the current UAV paths, are displayed on top of shaded terrain maps loaded from the DTED. When new data is received from a sensor, the sensor icon changes color, and the sensor data is displayed by clicking on the indicator.

The event controller is central to STOMP's simulation capability. It initializes the simulation, and provides several facilities for scripting simulation scenarios using a graphical interface. Scripting scenarios allows designers to study the reaction of the cooperation and path planning algorithms to various events. It also provides facilities for post-simulation analysis of network communications. As the simulation progresses, all state information of every element in the simulation (sensors and UAVs) is recorded, along with line-of-sight data computed from the Digital Terrain Elevation Data (DTED) data. This state information can be exported to a file and is suitable for postprocessing by other analysis tools.

The UAV and sensor objects contain state information and algorithms relevant to the simulation and operation of UAVs and sensors in the network. State updates may be obtained from the event controller (purely virtual), communication controller (purely real) or a combination (partially virtual). UAV objects are equipped with a flight controller, communication controller and automation algorithms. If the flight controller is to be connected to a real UAV, STOMP is connected, via a wireless serial interface, to an MP2000 autopilot engineered by Micropilot [12]. Using a communication wrapper, the MP2000 receives waypoint information from STOMP in order to control the direction of flight to service either virtual or real sensors. STOMP polls the MP2000 in order to update state information within the simulation asynchronously while the simulation is stepping through time.

The communications controller coordinates all communication between the main event controller, and real and virtual UAVs and sensors. Since this controller is the gateway to external hardware, it also routes command and control requests to the MP2000 autopilot via a wireless serial interface. The wireless Ethernet network in real sensors and UAVs is managed using Mobile Mesh [13] routing software using TCP sockets. STOMP operates at the application layer, while Mobile Mesh operates at layer 3 of the OSI model, thus STOMP remains independent of the specific communication and routing methods used in the real world environment.

During the first year of this project, a need arose for a small, flexible communications node. This project investigated several alternatives, and settled on the IEEE 802.11 standard coupled with a small Linux computer (see Figure 5). The ideas initiated in this project were adopted and refined in an Engineering Tech Base project, and several communications nodes were built. These nodes were used to conduct several flight tests using Laboratory UAVs. The objective of these experiments was to determine appropriate data transfer mechanisms. The experiments consisted of collecting imagery from a camera attached to a communications node, transferring the data to a UAV, and relaying the data to a ground station. It was found that the communication channel between the UAV and communication node was too unreliable for standard file transfer techniques. As a result of these experiments, a new protocol was devised that allows more robust file transfer. This protocol has been incorporated into STOMP, and it is suitable for a wide variety of UAV-to-sensor applications.

Although STOMP has been designed to send and receive any kind of data, the display controller was designed to display and allow the user to access image data. An illustration of a STOMP image display is shown in Figure 4. In this example, the UAV has flown over a sensor collecting imagery from a bridge. As the UAV uploads the image, it transmits it to STOMP where the marker for the sensor changes color indicating new data. The operator clicks on the sensor indicator and the image is displayed.

## 5. SUMMARY AND FUTURE DIRECTIONS

An architecture for the coordinated operation of a fleet of Unmanned Air Vehicles collecting data over a broad area was presented. The architecture is designed to adapt to a variety of fundamental events so that data collection continues in an optimal fashion. A simulation and operations environment, called STOMP, was developed to further the investigation into the use of UAVs for data collection and sensing.

With regards to future directions, two potential directions emerge. The first it to continue in the area of operations research by including additional constraints to the path planning algorithm, such as the arrival and departure windows for a UAVs at certain waypoints. In this regard, the use of tabu searches should be examined (cf. [5] and [6]). The second avenue of investigation is to develop algorithms that will determine where to sense, i.e., put down waypoints, so that the UAVs can collect more data to confirm a hypothesis, or get a better estimate of the state of nature.

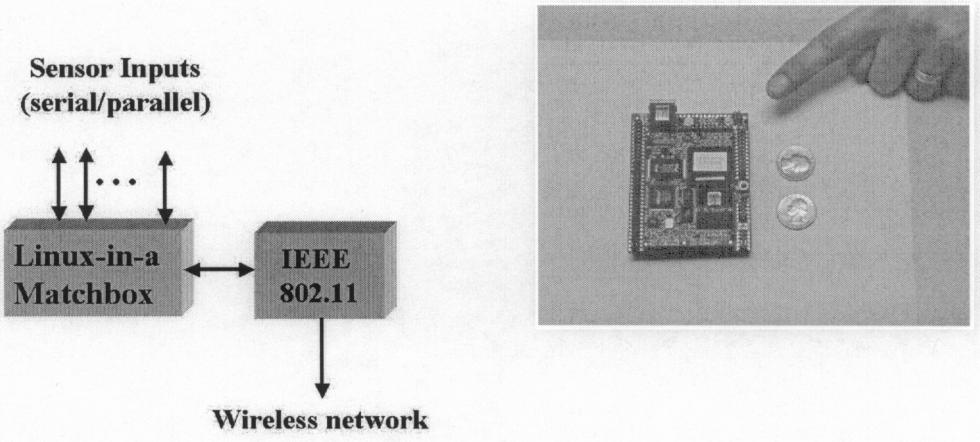


**Figure 5.** Block diagram and picture of a communications node

## 6. ACKNOWLEDGEMENTS

This research was performed under a grant from the Laboratory Directed Research and Development program, Lawrence Livermore National Laboratory. The authors would like to thank Dave McCallen, Director of the Center for Complex and Distributed Systems, and Don Meeker, Associate Director of Engineering, for their support of the project. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

## REFERENCES

1. C. T. Cunningham and R. S. Roberts. A adaptive path planning algorithm for cooperating unmanned air vehicles. In *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.
2. C. A. Kent and R. S. Roberts. Cooperation and path planning for unmanned air vehicles. Technical Report UCRL-JC-149915, Lawrence Livermore National Laboratory, September 2002.
3. E. D. Jones, R. S. Roberts, and T. C. Hsia. Stomp: A software architecture for the design and simulation of uav-based sensor networks. In *IEEE International Conference on Robotics and Automation*, Taipei, Tiawan, May 12-17 2003.
4. E. L. Lawler and et. al. *The Travelling Salesman Problem*. Wiley-Interscience, New York, 1985.
5. K. O'Rourke. Dynamic routing of unmanned aerial vehicles using reactive tabu search. In $67^{th}$ *Military Operations Research Symposium*, November 26 1999.
6. G. Barbarosoglu and D. Ozgur. A tabu search fro the vehicle routing problem. *Computers and Operations Research*, 26:255—270, 1999.
7. K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
8. J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1993.
9. R. S. Roberts and C. A. Kent. A method of self-adapting route planning for cooperating unmanned air vehicles. LLNL Record of Invention IL-11106, November 2002.
10. E. Gat. Three-layer architectures. In D. Kortnekamp, R. Peter Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robotics*. AAAI Press / MIT Press, Menlo Park, CA, 1998.
11. D. MacKenzie, R. Arkin, and J. Cameron. Multiagent mission specification and execution. *Autonomous Robots*, 10:29—52, 1997.

*This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.*

12. Micropilot Corporation. *MP2000 Autopilot*, 2001. Available online at: http://www.micropilot.com.
13. MITRE Corporation. *Providing Solutions for Mobile Adhoc Networking*, 2002. Available online at: http://www.mitre.org/tech_transfer/mobilemesh.